

# Semantic Web Modelling Languages (Part 1)

Tutorial at IJCAI2009  
July 13, 2009



Forschungszentrum Karlsruhe  
in der Helmholtz-Gemeinschaft



Universität Karlsruhe (TH)  
Research University • founded 1825

Pascal Hitzler



Markus Krötzsch



Sebastian Rudolph



AIFB, Universität Karlsruhe (TH)

<http://www.pascal-hitzler.de>

<http://korrekt.org>

<http://www.sebastian-rudolph.de>

# Outline

- **Introduction: The Quest for Semantics**
- **RDF**
- **RDF Schema**
- **Semantics for RDF(S)**
- **OWL Syntax and Semantics**

# Shortcomings of (Pure) XML

- Task: express "The Book 'Foundations of Semantic Web Technologies' is published at CRC Press."
- Many options:

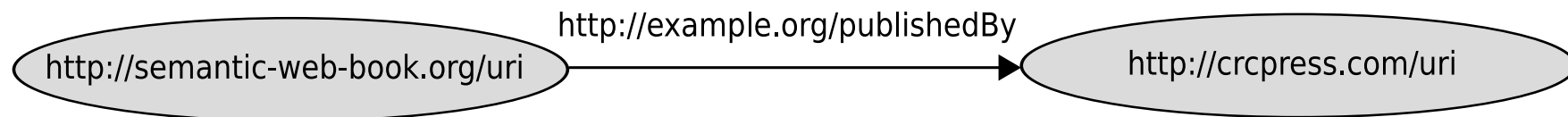
```
<published>  
<publisher>CRC Press</publisher>  
<book>Foundations of Semantic Web Technologies</book>  
</published>
```

```
<publisher name="CRC Press">  
<published book="Foundations of Semantic Web Technologies/>  
</publisher>
```

```
<book name="Foundations of Semantic Web Technologies">  
<published publisher="CRC Press"/>  
</book>
```

# RDF: Graphs instead of Trees

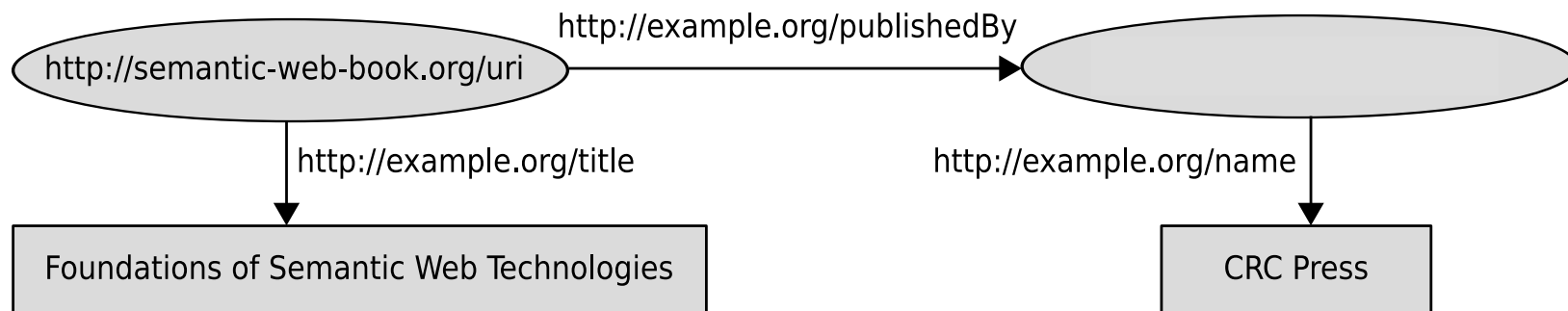
- **Solution: representation by directed graphs**



- **“Resource Description Framework”**
- **W3C Recommendation**  
(<http://www.w3.org/RDF>)
- **RDF is a data model (not one specific syntax)**
  - **originally designed for providing metadata for Web resources, later used for more general purposes**
  - **encodes structured informationen**
  - **universal machine-readable exchange format**

# Building blocks for RDF Graphs

- **URIs**
  - allow for referring to resources in an unambiguous way
- **literals**
  - describe data values (no separate existence in the world)
- **blank nodes (aka: empty nodes, bnodes)**
  - used to assert the existence of an object with certain characteristics without naming it



- **URI = Uniform Resource Identifier**
- **allow for denoting resources in a world-wide unambiguous way**
- **resources can be any object that possesses a clear identity (within the context of a given application)**
- **Examples: books, cities, humans, publishers, but also relations between those, abstract concepts, etc.**
- **already realized in some domains: e.g., ISBN for books**

# URIs - Syntax

- Builds on concept of URLs but not every URI refers to a Web document  
(but often the URL of a document is used as its URI)
- URI starts with so-called URI schema separated from the following part by ":" (e.g, **http**, **ftp**, **mailto**)
- mostly hierarchically organized

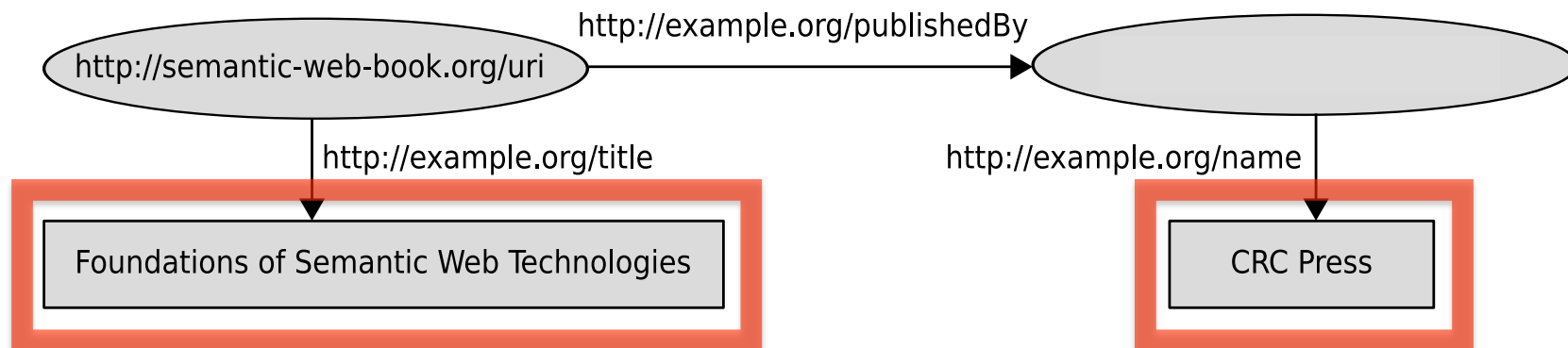


# Self-defined URIs

- necessary if no URI exists (yet) for a resource (or it is not known)
- strategy for avoiding unwanted clashes: use **http** URIs of webspace you control
- this also allows you to provide some documentation about the URI
- How to distinguish URI of a resource from URI of the associated documents describing it?
- Example: URI for "Othello"
  - don't use:  
**`http://de.wikipedia.org/wiki/Othello`**
  - rather use:  
**`http://de.wikipedia.org/wiki/Othello#URI`**

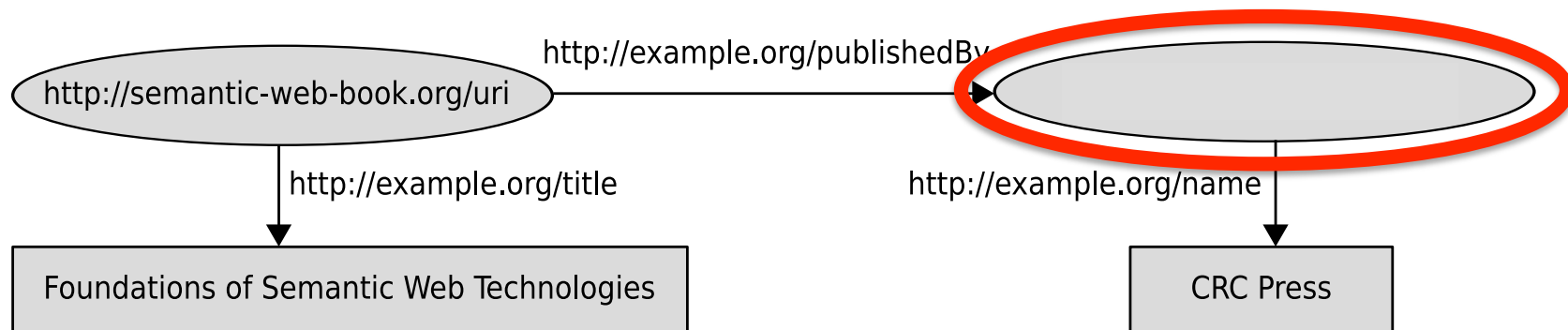
# Literals

- used for representing data values
- written down as strings
- interpreted via assigned *datatype*
- literals without explicitly associated datatype are treated as strings



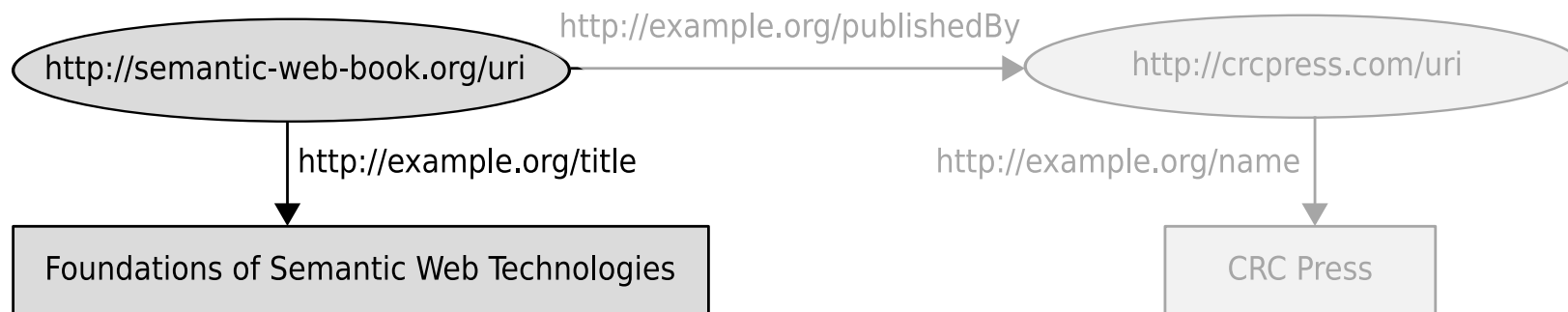
# Bnodes

- used to state existence of an entity the reference of which is not known



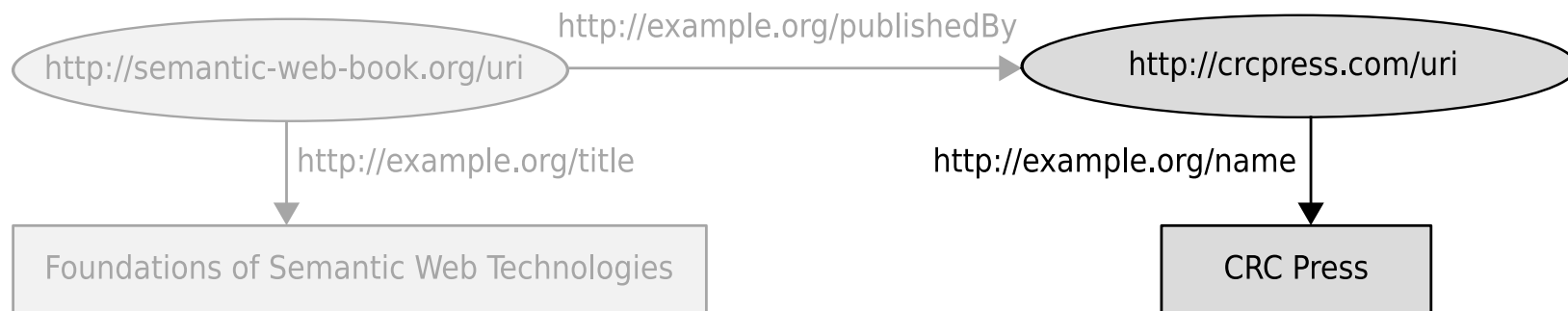
# Graphs as Triple Sets

- there are several ways for representing graphs
- in RDF we see graphs as set of vertex-edge-vertex triples



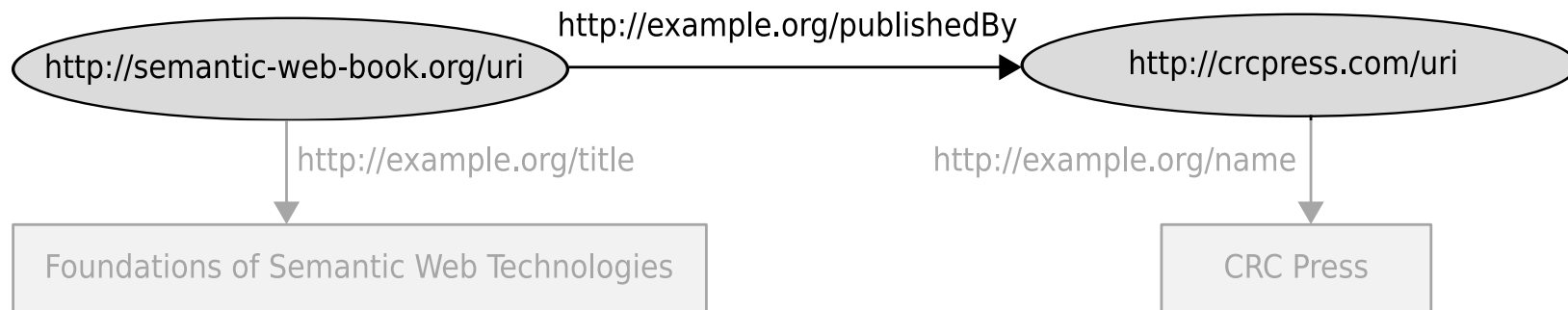
# Graphs as Triple Sets

- there are several ways for representing graphs
- in RDF we see graphs as set of vertex-edge-vertex triples



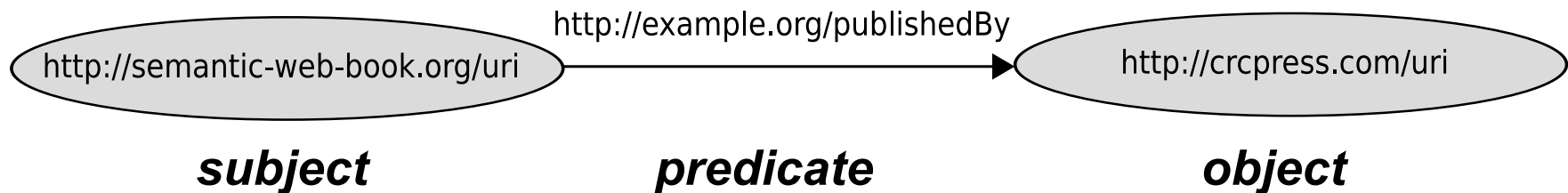
# Graphs as Triple Sets

- there are several ways for representing graphs
- in RDF we see graphs as set of vertex-edge-vertex triples



# RDF Triples

- **constituents of an RDF triple**



- **terms inspired by linguistics but doesn't always coincide**

- **eligible instantiations:**

**subject : URI or bnode**

**predicate : URI**

**objekt : URI or bnode or literal**

# Turtle - An Easy Syntax for RDF

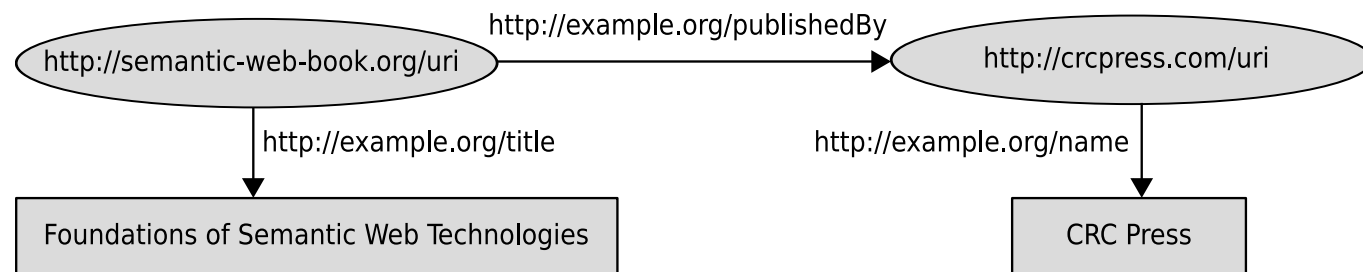
## Turtle notation:

- unabbreviated URIs in `<...>`
- literals in `"..."`
- period at the end of each triple
- extra spaces and linebreaks outside of names irrelevant

```
<http://semantic-web-book.org/uri> <http://example.org/publishedBy>  
    <http://crcpress.com/uri> .
```

```
<http://semantic-web-book.org/uri> <http://example.org/title>  
    "Foundations of Semantic Web Technologies" .
```

```
<http://crcpress.com/uri> <http://example.org/name> "CRC Press" .
```





# Turtle - An Easy Syntax for RDF

## Turtle notation:

- unabbreviated URIs in `<...>` but can be abbreviated by namespaces
- literals in `"..."`
- period at the end of each triple
- extra spaces and linebreaks outside of names irrelevant

@prefix book: <http://semantic-web-book.org/> .

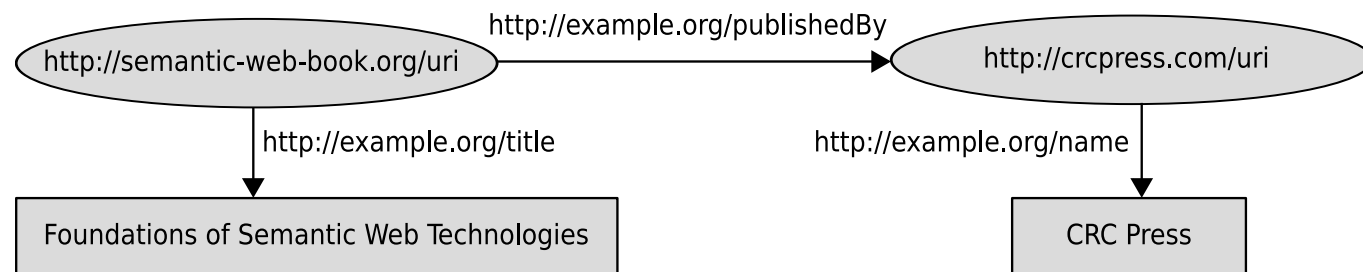
@prefix ex: <http://example.org/> .

@prefix crc: <http://crcpress.com/> .

book:uri ex:publishedBy crc:uri .

book:uri ex:title "Foundations of Semantic Web Technologies" .

crc:uri ex:name "CRC Press" .



# Turtle - An Easy Syntax for RDF

## Turtle notation:

- unabbreviated URIs in <...> but can be abbreviated by namespaces
- literals in “...”
- period at the end of each triple
- extra spaces and linebreaks outside of names irrelevant

@prefix book: <http://semantic-web-book.org/> .

@prefix ex: <http://example.org/> .

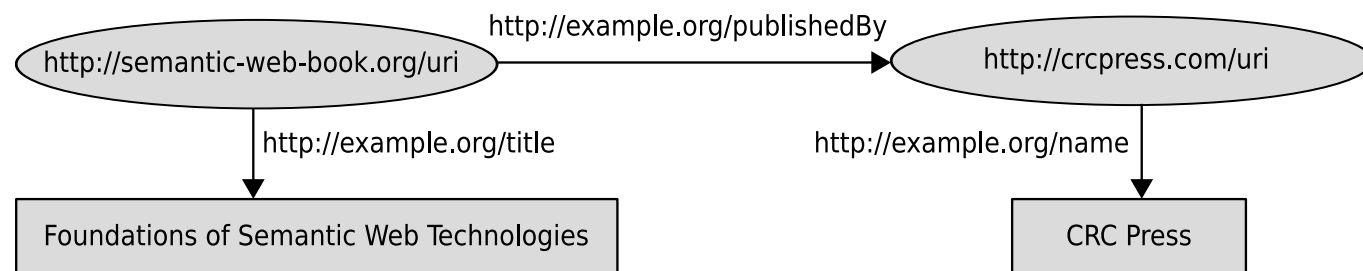
@prefix crc: <http://crcpress.com/> .

**repeated subjects may be left out**

book:uri      ex:publishedBy      crc:uri ;

                 ex:title                      "Foundations of Semantic Web Technologies" .

crc:uri      ex:name                      "CRC Press" .



# Turtle - An Easy Syntax for RDF

## Turtle notation:

- unabbreviated URIs in `<...>` but can be abbreviated by namespaces
- literals in `"..."`
- period at the end of each triple
- extra spaces and linebreaks outside of names irrelevant

`@prefix book: <http://semantic-web-book.org/> .`

`@prefix ex: <http://example.org/> .`

`@prefix crc: <http://crcpress.com/> .`

**repeated subjects may be left out**

`book:uri      ex:publishedBy      crc:uri ;`

`ex:title                      "Foundations of Semantic Web Technologies" ;`

`ex:author                    book:Hitzler, book:Krötzsch, book:Rudolph .`

`crc:uri      ex:name                      "CRC Press" .`

**several objects can be  
assigned to the same  
subject-predicate pairs**

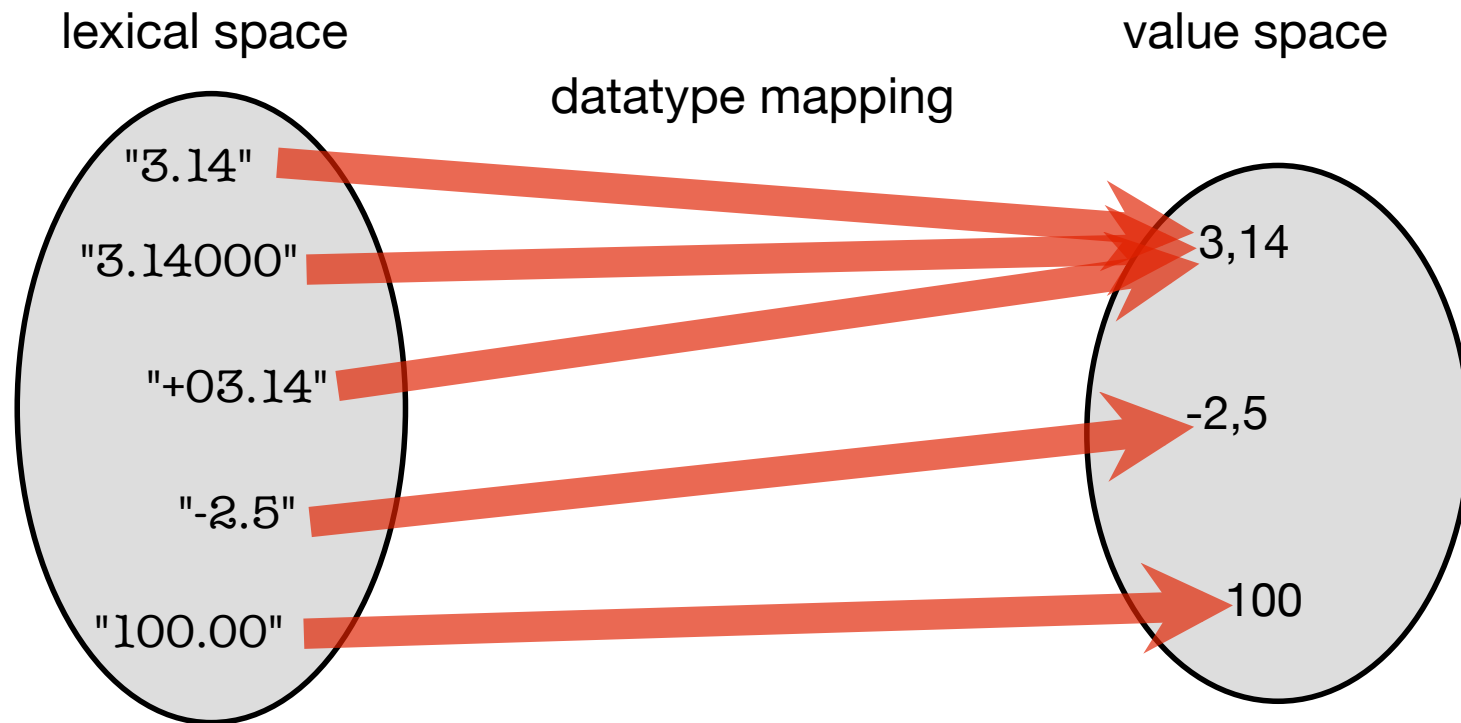
# XML-Syntax of RDF

- there is also an XML syntax for RDF
- it's for machines, so we don't deal with it here

```
<rdf:Description rdf:about="http://semantic-web-book.org/uri">  
  <ex:title>Foundations of Semantic Web Technologies</ex:title>  
  <ex:publishedBy>  
    <rdf:Description rdf:about="http://crcpress.com/uri">  
      <ex:name>CRC Press</ex:name>  
    </rdf:Description>  
  </ex:publishedBy>  
</rdf:Description>
```

# Datatypes – the Abstract View

## ■ Example: xsd:decimal



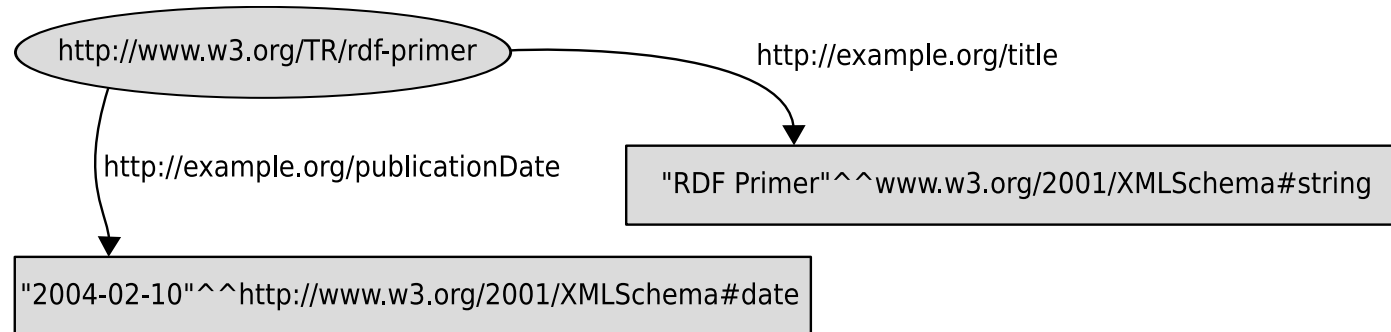
**"3.14" = "+03.14" holds for xsd:decimal but not for xsd:string**

# Datatypes in RDF

- by now: literals were untyped, interpreted as strings (i.e.: "02"<"100"<"11"<"2")
- typing literals with datatypes allows for more adequate (semantic = meaning-appropriate) treatment of values
- datatypes denoted by URIs and can be freely chosen
- frequently: xsd datatypes from XML
- syntax of typed literal:  
"datavalue"^^datatype-URI

# Datatypes in RDF – Example

## ■ Graph:



## ■ Turtle:

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .  
<http://www.w3.org/TR/rdf-primer>  
<http://example.org/title> "RDF Primer"^^xsd:string ;  
<http://example.org/publicationDate> "2004-02-10"^^xsd:date .
```

# The Predefined Datatype

- **rdf:XMLLiteral is the only datatype that is part of the RDF standard**
- **denotes arbitrary balanced XML “snippets”**



# Language Settings and Datatypes

- language settings only applicable to untyped literals

```
<http://www.w3.org/TR/rdf-primer> <http://example.org/title>  
  "Initiation à RDF"@fr, "RDF Primer"@en .
```

- distinct types or language settings – distinct literals

```
<http://crcpress.com/uri> <http://example.org/Name>  
  "CRC Press" ,  
  "CRC Press"@en ,  
  "CRC Press"^^xsd:string .
```

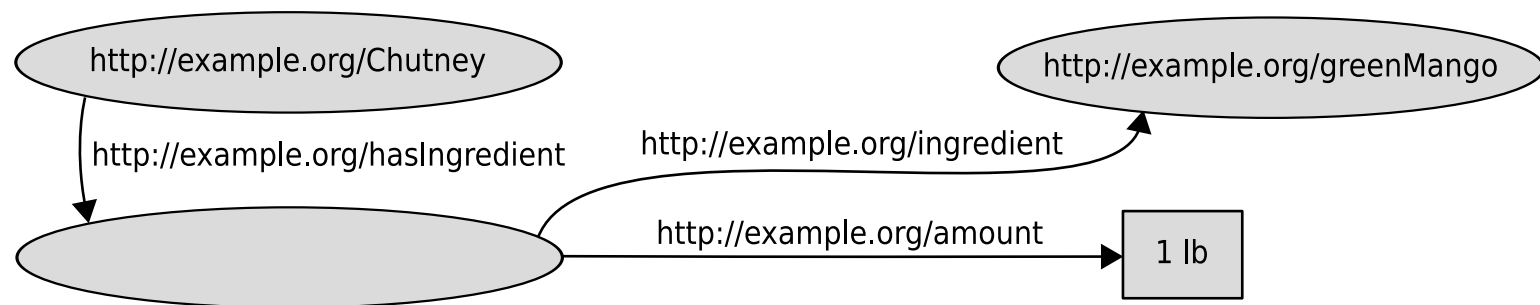
# n-ary Relationships

## ■ Cooking with RDF:

***“For the preparation of Chutney, we need the following:  
1 lb green mango, 1 tsp. Cayenne pepper, ...”***

dish	ingredient	amount
chutney	green mango	1 lb
chutney	cayenne pepper	1 tsp.

## ■ solved by auxiliary nodes (may be blank)



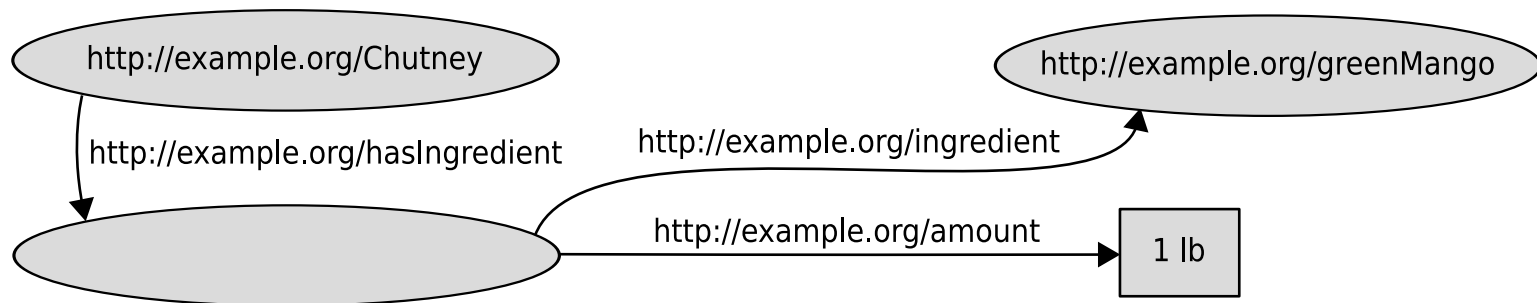
# n-ary Relationships

## ■ Turtle version 1:

```
@prefix ex: <http://example.org/> .  
ex:Chutney ex:hasIngredient _:idl .  
_:idl ex:ingredient ex:greenMango; ex:amount "1lb" .
```

## ■ Turtle version 2:

```
@prefix ex: <http://example.org/> .  
ex:Chutney ex:hasIngredient  
[ ex:ingredient ex:greenMango; ex:amount "1lb" ] .
```

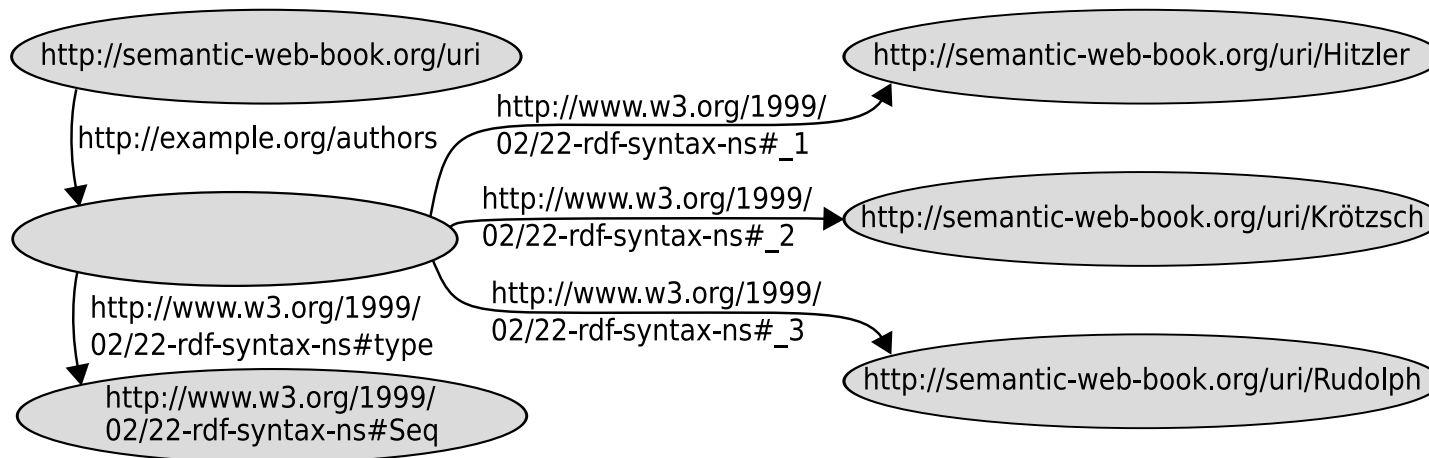


# Special Datastructures in RDF

- **open lists (containers)**
- **closed lists (collections)**
- **reified triples**

# Open Lists (Container)

## ■ Graph:

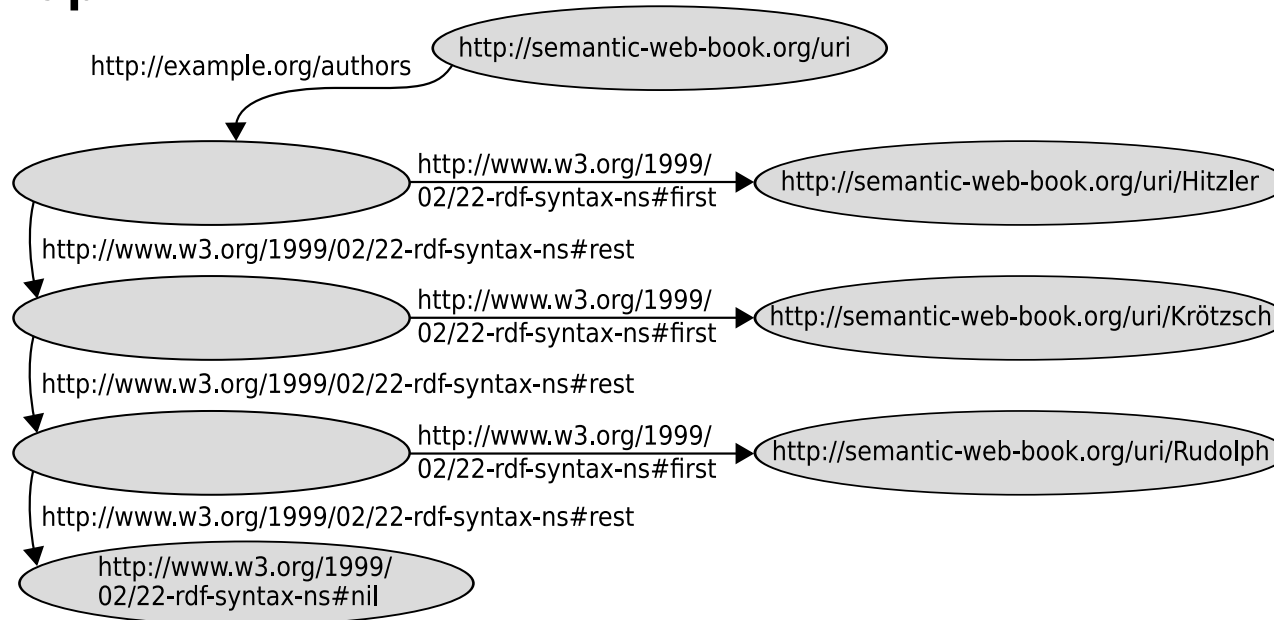


## ■ by `rdf:type` we assign a list type to the root node

- `rdf:Seq` – ordered list (sequence)
- `rdf:Bag` – unordered list
- `rdf:Alt` – set of alternatives or choices

# Closed Lists (Collections)

## ■ Graph:



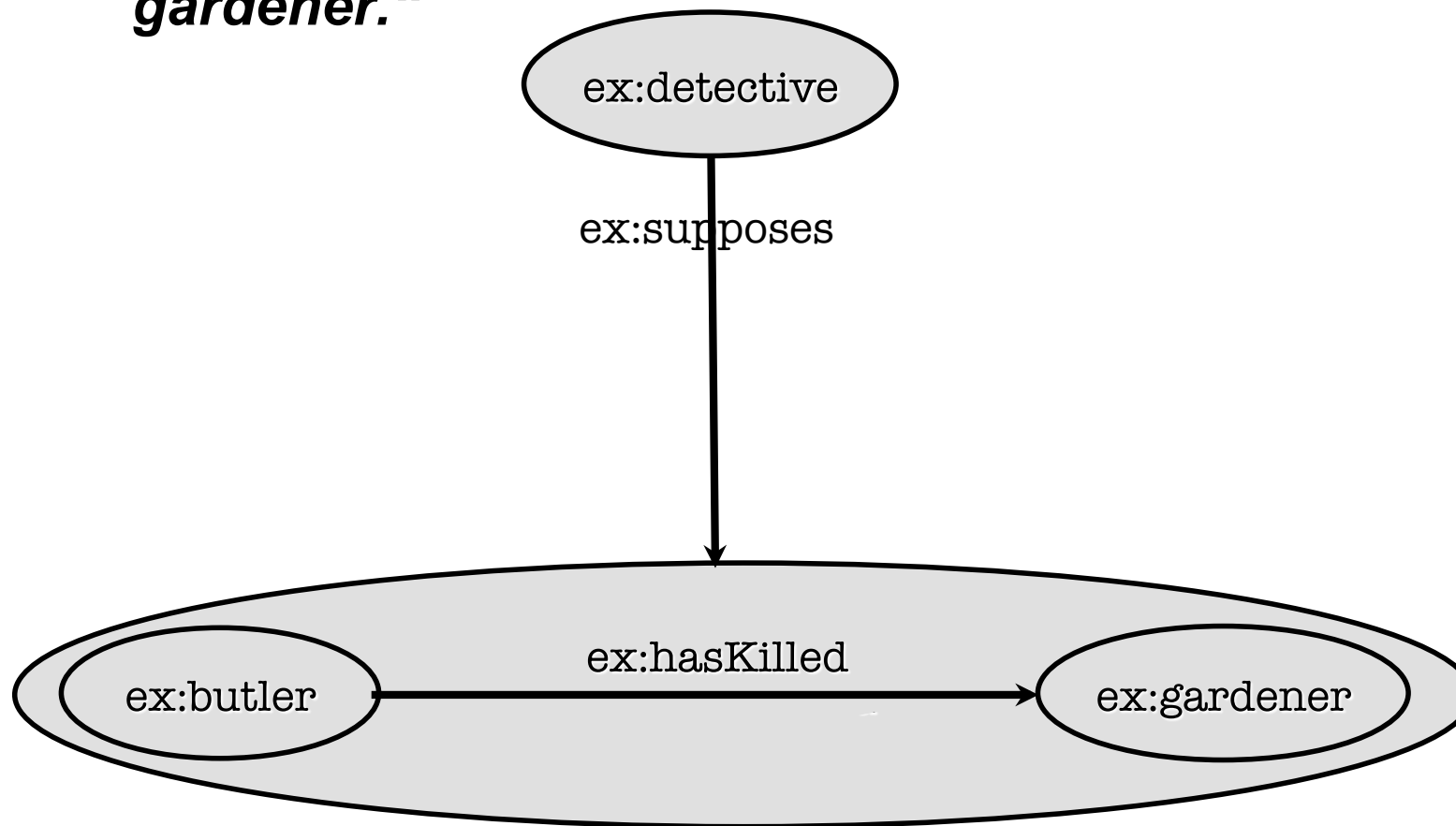
## ■ Abbreviation for Turtle:

@prefix book: <http://semantic-web-book.org/> .

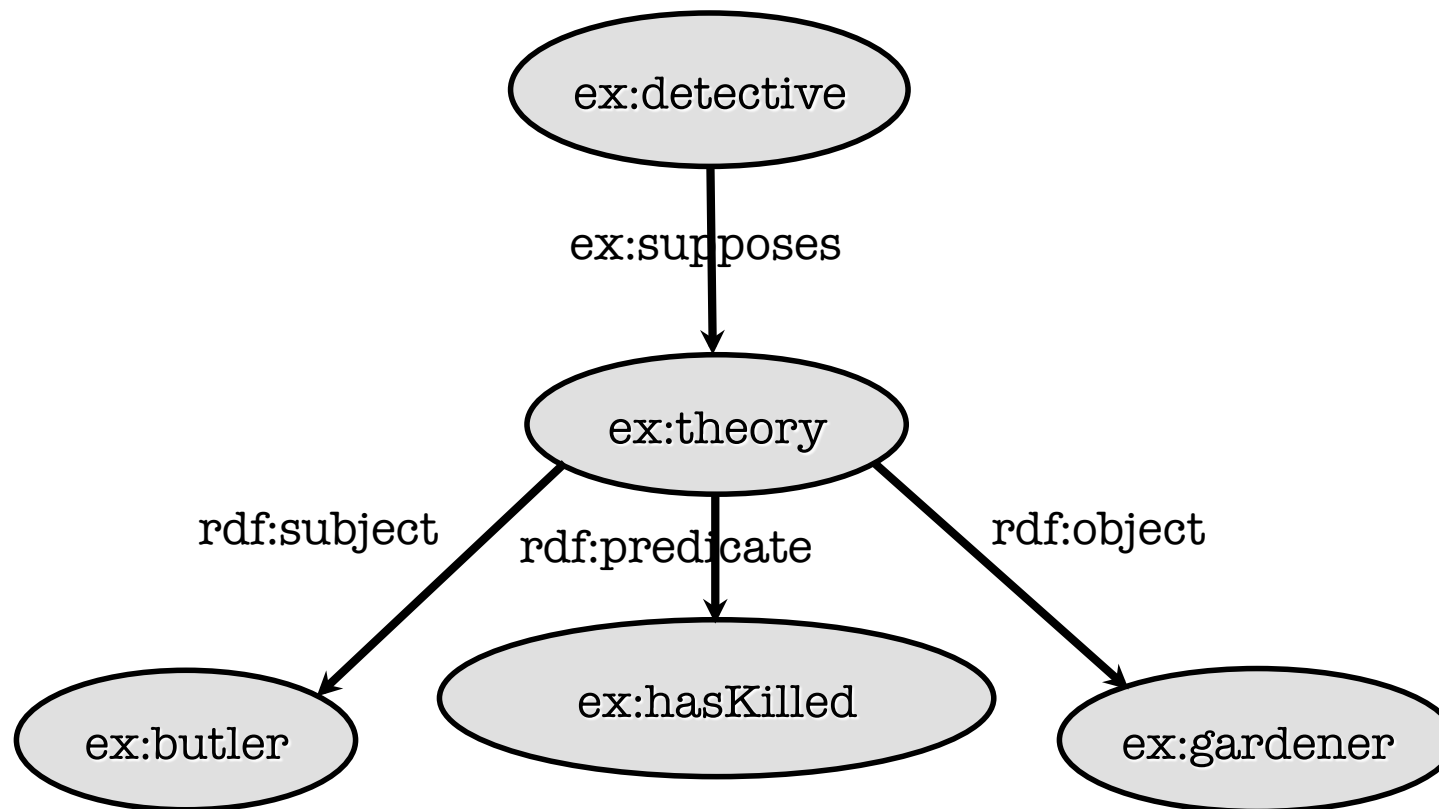
book:uri <http://example.org/authors>

( book:uri/Hitzler book:uri/Krötzsch book:uri/Rudolph ) .

- How to model propositions about propositions such as:  
*„The Detective supposes that the butler killed the gardener.“*



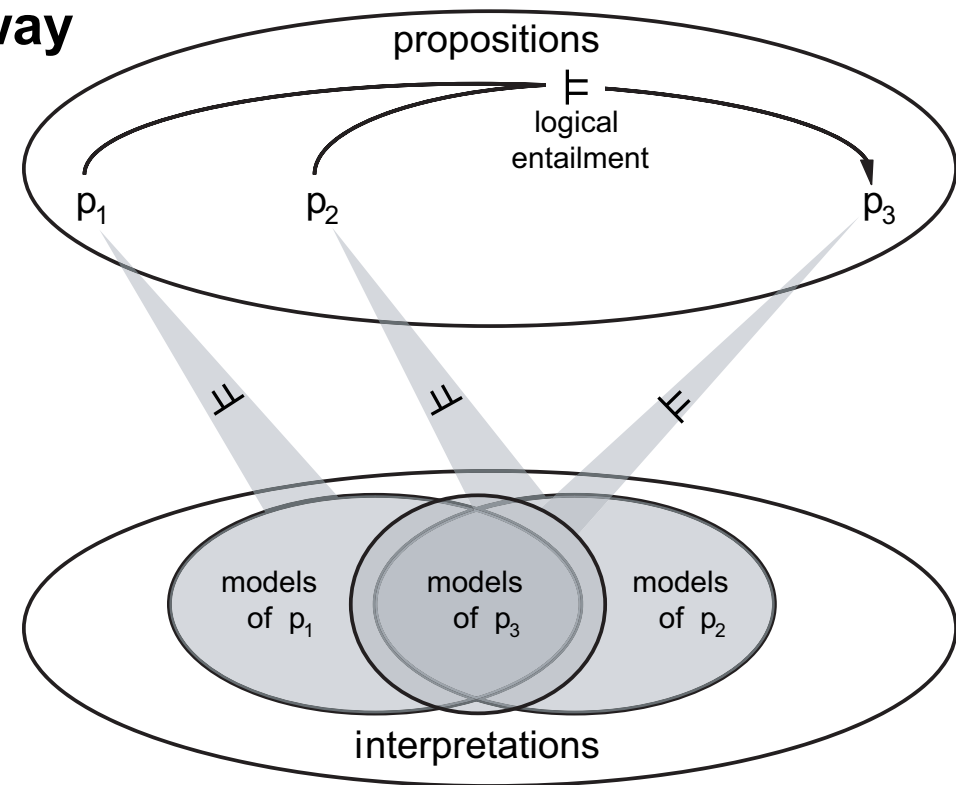
## ■ Solution: auxiliary node for nested proposition





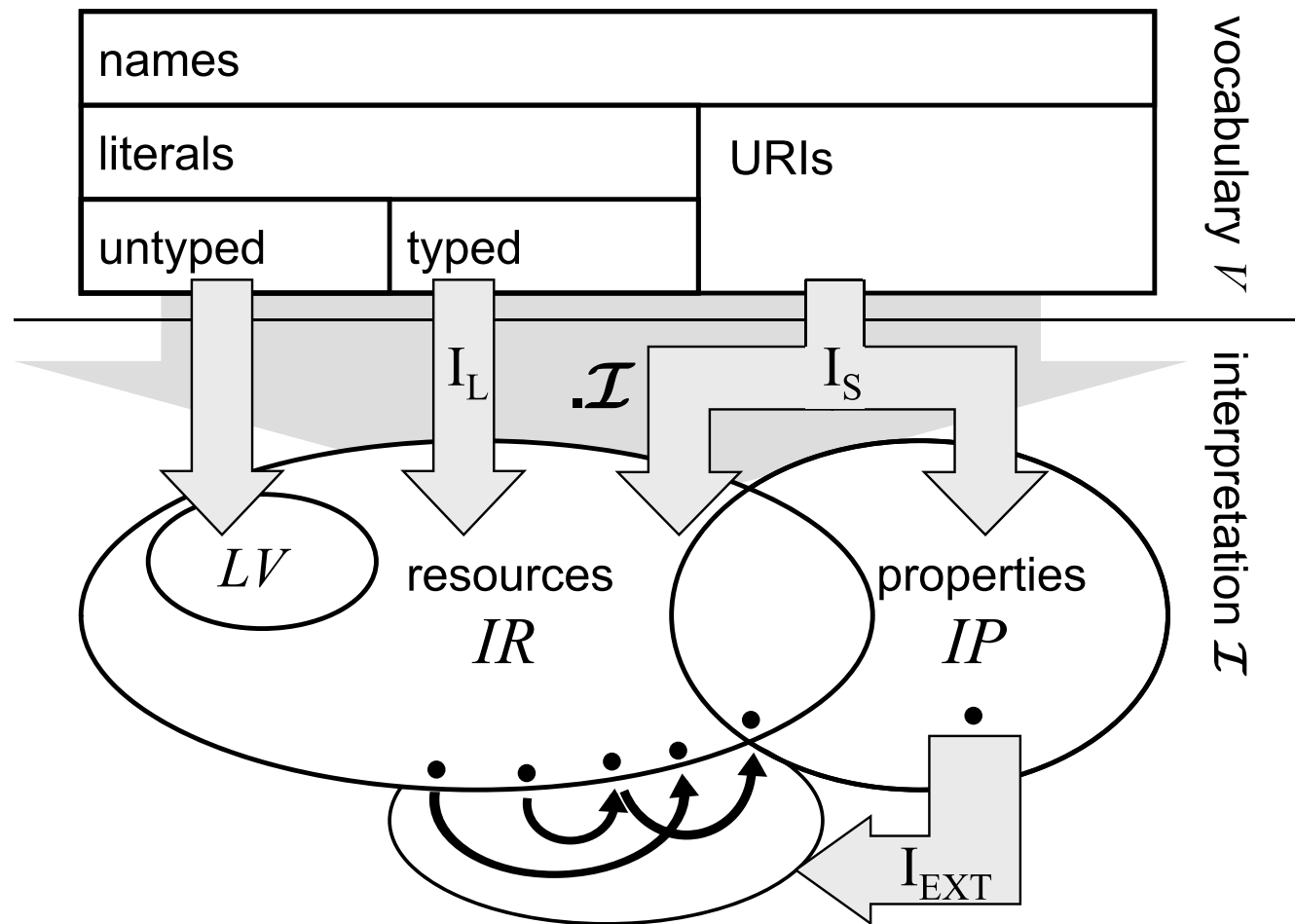
# Simple Semantics

- RDF is focused on information exchange and interoperability
- answers of RDF tools to entailment queries should coincide
- therefore, formal semantics needed
- defined in a model-theoretic way



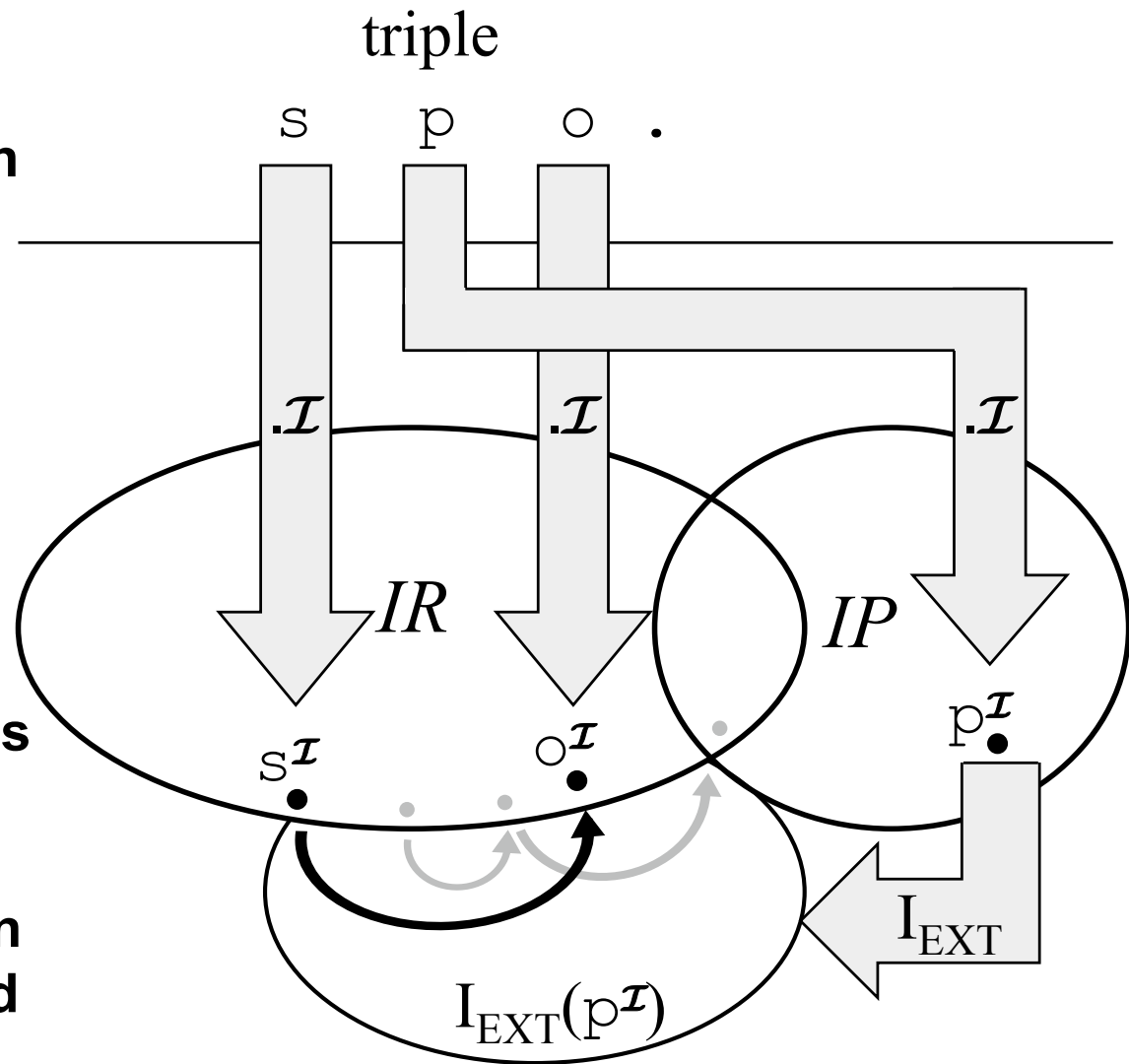
# Simple Semantics

## ■ Interpretation in RDF:



# Simple Semantics

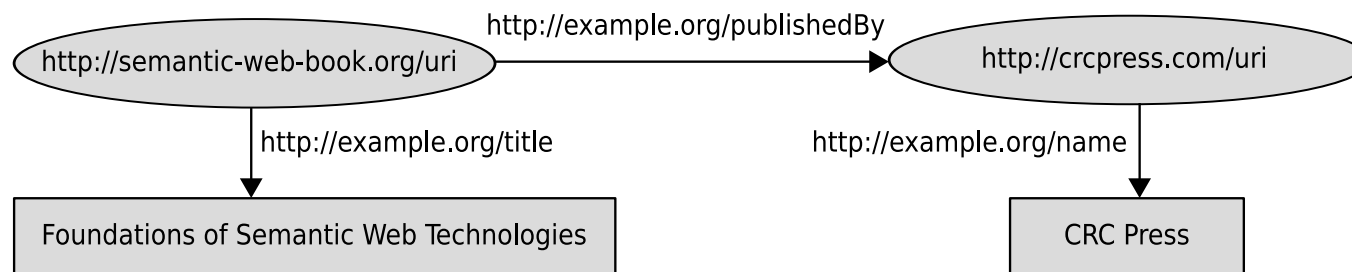
- when is a triple valid in an interpretation?
- a graph is valid, if all its triples are
- this settles the case for „grounded“ graphs
- graph with blank nodes is valid if they can be mapped to elements such that the condition on the right is satisfied



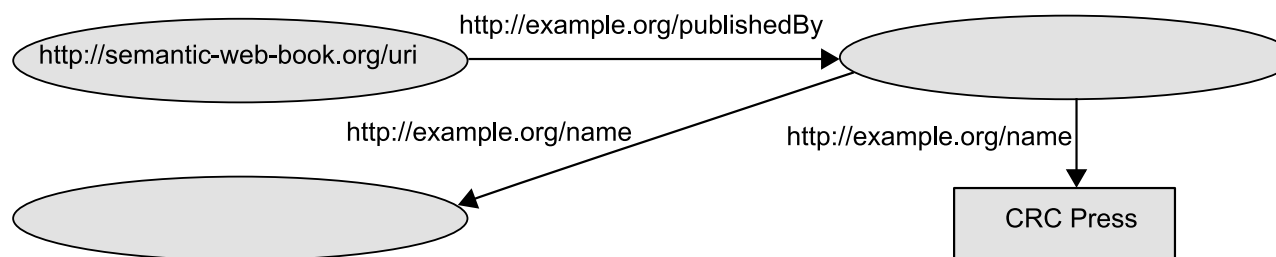
# Simple Entailment

- this model theory defines simple entailment
- this is essentially graph matching with bnodes being wildcards

## Example: the graph



## simply entails the graph



# Schema Knowledge with RDF(S)

- **RDF allows for specification of factual data**



- **= propositions about single resources (individuals) and their relationships**
- **desirable: propositions about generic groups of individuals, such as the class of publishers, of organizations, or of persons**
- **in database terminology: *schema knowledge***
- **RDF Schema (RDFS): part of the RDF W3C recommendation**

# Classes and Instances

```
book:uri rdf:type ex:Textbook .
```

- characterizes the specific book as an instance of the (self-defined) class of textbooks

- class-membership not exclusive:

```
book:uri rdf:type ex:Enjoyable .
```

- URIs can be typed as class-identifiers:

```
ex:Textbook rdf:type rdfs:Class .
```

# Subclasses

- we want to express that every textbook is a book, e.g., that every instance of the class `ex:Textbook` is “automatically” an instance of the class `ex:Book`
- realized by `rdfs:subClassOf` property:

`ex:Lehrbuch rdfs:subClassOf ex:Buch .`

- `rdfs:subClassOf` is defined to be transitive and reflexive
- rule of thumb:

`rdf:type` means  $\in$

`rdfs:subClassOf` means  $\subseteq$

- **technical term for Relations, Correspondencies**
- **Property names usually occur in predicate position in factoid RDF triples**
- **characterize, how two resources are related**
- **mathematically: set of pairs:**  
**married\_with = {(Adam,Eva),(Brad,Angelina),...}**
- **URI can be marked as property name by typing it accordingly:**

`ex:publishedBy rdf:type rdf:Property .`



# Subproperties

- in analogy to subclass relationships
- representation in RDFS via `rdfs:subPropertyOf` e.g.:  
`ex:happilyMarriedWith rdfs:subPropertyOf rdf:marriedWith .`
- then, given  
`ex:Markus ex:happilyMarriedWith ex:Anja .`  
**we can deduce**  
`ex:Markus ex:marriedWith ex:Anja .`

# Property Restrictions

- **properties may give hints what types the linked resources have, e.g. we know that `ex:publishedBy` connects publications with publishers**
- **i.e., for all URIs `a`, `b` where we know**  
`a ex:publishedBy b .`  
**we want to automatically follow:**  
`a rdf:type ex:Publication .`  
`b rdf:type ex:Publisher .`
- **this generic correspondence can be encoded in RDFS:**  
`ex:publishedBy rdfs:domain ex:Publication .`  
`ex:publishedBy rdfs:range ex:Publisher .`

# Einschränkung von Property

- with property restrictions, semantic interdependencies between properties and classes can be specified
- **Caution: property restrictions are interpreted globally and conjunctively, e.g.**

```
ex:authorOf rdfs:range ex:Cookbook .  
ex:authorOf rdfs:range ex:Storybook .
```

**means: everything which is authored by somebody is both a cookbook and a storybook**

- **thus: always use most generic classes for domain/range statements**

# Additional Information

- **used to add human-readable information (comments or names)**
- **for compatibility reasons graph-based representation recommended; set of properties for that purpose:**
  - **`rdfs:label` assigns an alternative name (encoded as literal) to an arbitrary resource**
  - **`rdfs:comment` assigns a more comprehensive comment (also literal)**
  - **`rdfs:seeAlso`, `rdfs:definedBy` refer to resources (URIs!) containing further information about the subject resource**

- **RDFS interpretations take care of RDF(S)-specific vocabulary by imposing additional conditions on simple interpretations:**
  - **all URIs and bnodes are of type `rdf:Resource`**
  - **triple predicates are of type `rdf:Property`**
  - **all well-typed and untyped literals are of type `rdf:Literal`**
  - **types of triple subjects/objects correspond to `rdfs:domain/` `rdfs:range` statements**
  - **`rdfs:subClassOf` and `rdfs:subPropertyOf` are interpreted reflexive and transitive and “inheriting”**
  - **well-formed XML-Literals are mapped into LV, ill-formed ones go somewhere else**
  - **...and many more**

# RDFS Entailment – Automation

- RDFS entailment can be decided via rule-like deduction calculus (NP-complete)

$$\begin{array}{c}
 \frac{}{u \text{ a } x} \text{ rdfsax} \qquad \frac{u \text{ rdfs:subPropertyOf } v . \quad v \text{ rdfs:subPropertyOf } x .}{u \text{ rdfs:subPropertyOf } x .} \text{ rdfs5} \qquad \frac{u \text{ rdf:type rdfs:ContainerMembershipProperty .}}{u \text{ rdfs:subPropertyOf rdfs:member .}} \text{ rdfs12} \\
 \\
 \frac{u \text{ a } \_ : n .}{u \text{ a } l .} \text{ gl} \qquad \frac{u \text{ rdf:type rdf:Property .}}{u \text{ rdfs:subPropertyOf } u .} \text{ rdfs6} \qquad \frac{u \text{ rdf:type rdfs:Datatype .}}{u \text{ rdfs:subClassOf rdfs:Literal .}} \text{ rdfs13} \\
 \\
 \frac{u \text{ a } l .}{\_ : n \text{ rdf:type rdfs:Literal .}} \text{ rdfs1} \qquad \frac{a \text{ rdfs:subPropertyOf } b . \quad u \text{ a } y .}{u \text{ b } y .} \text{ rdfs7} \\
 \\
 \frac{a \text{ rdfs:domain } x . \quad u \text{ a } y .}{u \text{ rdf:type } x .} \text{ rdfs2} \qquad \frac{u \text{ rdf:type rdfs:Class .}}{u \text{ rdfs:subClassOf rdfs:Resource .}} \text{ rdfs8} \\
 \\
 \frac{a \text{ rdfs:range } x . \quad u \text{ a } v .}{v \text{ rdf:type } x .} \text{ rdfs3} \qquad \frac{u \text{ rdfs:subClassOf } x . \quad v \text{ rdf:type } u .}{v \text{ rdf:type } x .} \text{ rdfs9} \\
 \\
 \frac{u \text{ a } x .}{u \text{ rdf:type rdfs:Resource .}} \text{ rdfs4a} \qquad \frac{u \text{ rdf:type rdfs:Class .}}{u \text{ rdfs:subClassOf } u .} \text{ rdfs10} \\
 \\
 \frac{u \text{ a } v .}{v \text{ rdf:type rdfs:Resource .}} \text{ rdfs4b} \qquad \frac{u \text{ rdfs:subClassOf } v . \quad v \text{ rdfs:subClassOf } x .}{u \text{ rdfs:subClassOf } x .} \text{ rdfs11}
 \end{array}$$

# Deployment of RDF

- today there is a variety of RDF tools
- software libraries for virtually every programming language
- freely available systems for handling large sets of RDF data (so-called RDF stores or triple stores)
- increasingly supported by commercial actors (e.g. Oracle)
- basis for several data formats: RSS 1.0, XMP (Adobe), SVG (vector graphics format)

# RDF(S) as Ontology Language?

- **RDFS language features allow for modeling certain semantic aspects of a domain of interest**
- **hence, RDFS can be seen as a *lightweight* ontology language**



# RDF(S) as Ontology Language?

## Shortcomings of RDF(S):

### ■ “weak” semantics:

```
ex:speaksWith rdfs:domain    ex:Homo .  
ex:Homo      rdfs:subClassOf ex:Primates .
```

**does not entail**

```
ex:speaksWith rdfs:domain    ex:Primates .
```

### ■ expressivity: no negative information can be specified, no cardinality, no disjunction...

**Thanks!**

**[http://semantic-web-grundlagen.de/wiki/ESWC09\\_Tutorial](http://semantic-web-grundlagen.de/wiki/ESWC09_Tutorial)**